VPro™ Odyssey Graphics Diagnostic Reference

SGI Confidential & Proprietary Information - For Internal Recipients Only

Document Number 108-0314-003

CONTRIBUTORS
Written by Jim Ostrom.
Revised by Darrin Goss.
Edited by Cindi Leiser.
Production by Rhonda Kunsman.
Engineering contributions by Gregoire Banderet and Jason Godfrey.

Record of Revision

Version	Description
001	February 2001 Original printing.
002	November 2001 Revised to include support for the SGI 3000 family VPro V12 configuration (V brick) and the CMAP patterns test.
003	April 2002 Revised to include support for Silicon Graphics Fuel visual workstations.

Contents

Tables

Examples

VPro Odyssey Graphics Diagnostic Reference

This document describes the VPro™ Odyssey graphics diagnostic tests and utilities. You can use these tests and utilities to verify operation of the VPro graphics hardware in SGI® Onyx® 3000 series systems with InfinitePerformance™ graphics, Silicon Graphics Fuel™ visual workstations, and Silicon Graphics® Octane2™ visual workstations. The Odyssey graphics tests and utilities run under IRIX® version 6.5.9 or later.

Odyssey Graphics Hardware

The VPro Odyssey graphics subsystem consists of an SGI proprietary chip set and associated software. The chip set includes the buzz ASIC, pixel blaster and jammer (PB&J) ASIC, and associated SDRAM.

The buzz ASIC is a single-chip graphics pipeline. It operates at 266 MHz and contains on-chip SRAM. The buzz ASIC interfaces with:

- SDRAM (32 Mbytes or 128 Mbytes) that operates at 133 MHz
- PB&J ASIC via a dedicated bus
- Other system hardware using a 16-bit, 400-MHz peer-to-peer XIO interface

The PB&J ASIC is the display backend chip for the graphics subsystem.

Running the Odyssey Diagnostics

You **must** run the Odyssey diagnostics from a dumb terminal or serial port console that is connected to the system or over a network using a remote login. No graphical user applications should be running on a system when you run the Odyssey diagnostics.

If you want to run a predefined suite of Odyssey diagnostic tests to verify operation of the graphics hardware, refer to “Running the Odyssey Diagnostics Automatically” on page 2.

If you want to run specific Odyssey diagnostic tests or utilities, refer to “Running the Odyssey Diagnostics Manually” on page 5.

Running the Odyssey Diagnostics Automatically

You can run a predefined suite of Odyssey diagnostics automatically with the `/usr/diags/ody/run.pl` perl script. (Install the `field_diags_odyssey` IRIX `inst` package included in the Internal Support Tools CD releases to load the `run.pl` script on a system.)

The `run.pl` script performs the following operations:

1. Stops the X Window System (terminates the X Display Manager)
2. Runs `./odydiags -reset`
3. Runs `./odydiags -registers -sdram -dcb -screen_width 1280 -screen_height 1024 LOG=tmp.log`
4. Runs `./odydiags -i2c LOG=tmp.log`
5. Runs `./odydiags -cmap_data_bus -cmap_addr_bus LOG=tmp.log`
6. Runs `./odydiags -load_cmap -raster LOG=tmp.log`
7. Runs `./odydiags -load_cmap -shader LOG=tmp.log`

The `run.pl` script takes two optional arguments:

```
# run.pl [loops] [-<optional test list>]
# run.pl 10 -all
```

The first argument specifies a loop count, and the second argument specifies a test list. If you specify a loop count of 0, the test loops continuously. The default loop count is 1.

Table 1 describes the log files that are created by the `run.pl` script.

Table 1 run.pl Script Log Files

Log File	Description
`.tmp.log`	A temporary diagnostic harness log file.
`.details.log`	A concatenation of all harness log files (copy of the output).
`.summary.log`	A log file created for each pass.
`.loopsummary.log`	A summary log file of multiple passes.

To run the Odyssey diagnostics automatically on a system running IRIX 6.5.9 or IRIX 6.5.10, refer to “Running the Odyssey Diagnostics Automatically on IRIX 6.5.9 or IRIX 6.5.10” on page 3.

To run the Odyssey diagnostics automatically on a system running IRIX 6.5.11 or later, refer to “Running the Odyssey Diagnostics Automatically on IRIX 6.5.11 and Later” on page 4.

Running the Odyssey Diagnostics Automatically on IRIX 6.5.9 or IRIX 6.5.10

If a system is running IRIX 6.5.9 or 6.5.10, you must configure it to use the data console before you start testing the system. Perform the following procedure to run the Odyssey diagnostics automatically with the `run.pl` script on a system that is running IRIX 6.5.9 or IRIX 6.5.10:

1. Log in as root and set the console to the terminal that is connected to the first serial port (or the L1 port) by using one of the following commands:

   # **nvram console d**

2. Enter the following command to disable the X Window System:

   # **chkconfig windowsystem off**

3. Reboot the system with these settings to run Odyssey diagnostics.
4. Log in as root from the system console that is connected to the first serial port (or the L1 port) or from a network connection.
5. Enter the following command to change to the directory that contains the Odyssey diagnostics:

   # **cd /usr/diags/ody**

6. Enter the following command to run the Odyssey diagnostics automatically:

   # **./run.pl**

7. After you have completed testing the system, enter the following command to select the graphic console:

   # **nvram console g**

8. Enter the following command to enable the window system:

   # **chkconfig windowsystem on**

9. Reboot the system.

Running the Odyssey Diagnostics Automatically on IRIX 6.5.11 and Later

If a system is running IRIX 6.5.11 or later, you can run `odydiags` while the graphic console is enabled. This enables you to run diagnostics without rebooting the system. Perform the following procedure to run the Odyssey diagnostics automatically with the `run.pl` script on a system that is running IRIX 6.5.11 or later:

1. Log in as root from the system console that is connected to the first serial port (or the L1 port) or from a network connection.
2. Enter the following command to change to the directory that contains the Odyssey diagnostics:

   ```
   # cd /usr/diags/ody
   ```

3. Enter the following command to run the Odyssey diagnostics automatically:

   ```
   # ./run.pl
   ```

4. After you have completed testing the system, reboot the system.

Running the Odyssey Diagnostics Manually

You can also manually run specific Odyssey diagnostic tests and utilities:

- To run the Odyssey diagnostics manually on a system running IRIX 6.5.9 or IRIX 6.5.10, refer to “Running the Odyssey Diagnostics Manually on IRIX 6.5.9 or IRIX 6.5.10” on page 5
- To run the Odyssey diagnostics manually on a system running IRIX 6.5.11 or later, refer to “Running the Odyssey Diagnostics Manually on IRIX 6.5.11 or Later” on page 6.

Running the Odyssey Diagnostics Manually on IRIX 6.5.9 or IRIX 6.5.10

If a system is running IRIX 6.5.9 or 6.5.10, you must configure it to use the data console before you start testing the system. Perform the following procedure to run Odyssey diagnostics manually on a system that is running IRIX 6.5.9 and IRIX 6.5.10:

1. Log in as root and set the console to the terminal that is connected to the first serial port (or the L1 port) by using one of the following commands:

   ```
   # nvram console d
   ```

2. Enter the following command to disable the X Window System:

   ```
   # chkconfig windowsystem off
   ```

3. Reboot the system with these settings to run Odyssey diagnostics.
4. Log in as root from the system console that is connected to the first serial port (or the L1 port) or from a network connection.
5. Enter the following command to change to the directory that contains the Odyssey diagnostics:

   ```
   # cd /usr/diags/ody
   ```

6. Run the desired Odyssey diagnostic tests or utilities with the `odydiags` command.
7. After you have completed testing the system, enter the following command to select the graphical console:

   ```
   # nvram console g
   ```

8. Enter the following command to enable the window system:

   ```
   # chkconfig windowsystem on
   ```

9. Reboot the system.

Running the Odyssey Diagnostics Manually on IRIX 6.5.11 or Later

If a system is running IRIX 6.5.11 or later, you can run `odydiags` while the graphics console is enabled. This enables you to run diagnostics without rebooting the system. Perform the following procedure to run Odyssey diagnostics manually on a system that is running IRIX 6.5.11 or later:

1. Log in as root from the system console that is connected to the first serial port (or the L1 port) or from a network connection.
2. Run the desired Odyssey diagnostic tests or utilities with the `odydiags` command.
3. Enter the following command to change to the directory that contains the Odyssey diagnostics:

   ```
   # cd /usr/diags/ody
   ```

4. After you have completed testing the system, enter the following command to reset the graphics hardware:

   ```
   # odydiags -reset
   ```

5. Enter the following command to restart the window system:

   ```
   # /usr/gfx/startgfx
   ```

Common Command-line Options

Table 2 lists command-line options that are common to all Odyssey diagnostics. Use these options to modify test behavior.

Note: The individual test descriptions describe the command-line options that are specific to each test.

Table 2 Common Odyssey Diagnostic Command-line Options

Option	Description
`--<`*test_name*`>`	Prevents the specified test from running.
`-all`	Runs all standard tests. **Does not function correctly at the time this document was printed. Do not use this option; use the `run.pl` script to run all of the Odyssey diagnostics.**
`-color`	Highlights `PASS` messages in green, `FAIL` messages in red, and unresolved error messages in yellow. This is the default.
`-config <`*filename*`>`	Loads the specified configuration file.
`-diag`	Displays `DIAG` messages. This is the default.
`-forever`	Runs the diagnostic indefinitely.
`-h|-help`	Runs with an interactive help menu and causes all other command-line options to be ignored. No tests will be run.
`-hrtb`	Displays `HRTB` messages. This is the default.
`-hwdebug <`*level*`>`	Specifies the verbosity of hardware debugging information. Valid values are 0 through 5; the default is 0.
`-info`	Displays `INFO` messages. This is the default.
`-interact`	Runs the test interactively.
`-loop`	Displays `LOOP` messages. This is the default.
`-meta`	Displays `META` messages. This is the default.
`-nocode`	Does not display `CODE` messages.
`-nocolor`	Does not highlight `PASS`, `FAIL`, and unresolved error messages in different colors.
`-nodiag`	Does not display `DIAG` messages.
`-noESP`	Disables logging of diagnostic events to Embedded Support Partner.
`-nohrtb`	Does not display `HRTB` messages. This is the default.
`-noinfo`	Does not display `INFO` messages.
`-noloop`	Does not display `LOOP` messages.
`-nometa`	Does not display `META` messages.
`-nonrequired`	Does not automatically select tests that are required by other tests being run.

Table 2 (continued) Common Odyssey Diagnostic Command-line Options

Option	Description
`-nonstd`	Runs all nonstandard tests.
`-norev`	Does not display `REV` messages.
`-norslt`	Does not display `RSLT` messages.
`-notest`	Does not display `TEST` messages.
`-notime`	Does not display `TIME` messages. This is the default.
`-notrace`	Does not display `TRCE` messages.
`-operator`	Provides minimal output to the screen. Limits messages to `CMDL`, `META`, `LOOP`, `REV`, `RSLT`, and `****ERROR` messages.
`-rev`	Displays `REV` messages. This is the default.
`-rslt`	Displays `RSLT` messages. This is the default.
`-runtime <`*time*`>`	Runs the test for the specified time (in minutes).
`-test`	Displays `TEST` messages. This is the default.
`-time`	Displays `TIME` messages.
`-trace`	Displays `TRCE` messages. This is the default.
<*test_name*>=<*number*>	Runs the specified test for the specified number of times.
`HWDEBUG=<`*level*`>`	Same as `-hwdebug <`*level*`>`.
`INDENT_STEP=<`*step*`>`	Indents the text by the specified number of spaces.
`LOG=<`*filename*`>`	Copies diagnostic output to the specified file.
`MAXERR=<`*number*`>`	Exits the diagnostic after the specified number of tests have failed. The default is 1.
`MAX_ERRORS=<`*number*`>`	Exits the diagnostic after the specified number of errors have occurred. The default is 20.
`META=<`*number*`>`	Prints out `META` information when the specified number of loops is completed.
`REPEAT=<`*loops*`>`	Runs the list of tests the specified number of times. The default is 1.
`TRACE=<`*filename*`>`	Logs `TRCE` messages to the specified file.
`WIDTH=<`*number*`>`	Sets the width of the diagnostic messages to the specified number; does not include the output tag. The default is 59.

Common Output

Odyssey diagnostics begin each line of output with common output tags. These output tags make it easier to interpret the information that the test displays. Refer to Table 3 for a listing of all the output tags and their descriptions.

Odyssey diagnostics display similar pass and fail output. Messages that indicate that a test has passed successfully are highlighted in green; messages that indicate that a test has failed are highlighted in red; and messages that indicate that a test did not complete or was unresolved are highlighted in yellow.

Table 3 Common Odyssey Diagnostic Output Tags

Tag	Description
ABRT	Indicates an error that caused the diagnostic to unexpectedly exit.
BRD#	Displays the board number.
CMDL	Displays the command line that is used to start the diagnostic.
CODE	Calls out a specific board or chip.
DIAG	Displays information about the cause of a failure.
HDBG	Displays information that is useful for debugging hardware.
HRTB	Indicates that the diagnostic is still running during time-consuming operations.
INFO	Displays general information about the hardware or diagnostic operations.
LOOP	Signals the end of a loop.
NOTE	Displays important diagnostic information.
REV	Displays the revision level of the diagnostic.
RSLT	Displays the result of the most recent test. Highlighted green for pass, red for fail, and yellow for unresolved.
TEST	Indicates the start of a test.
TIME	Displays the current time.
TOUT	Appears when the diagnostic exits because it exceeds the maximum run time.
TRCE	Indicates (traces) the code that is used when a test fails; this should not be used often by field engineers.
****ERROR	Signals that a hardware error was detected.

Table 4 lists the main `odydiags` tests, provides a brief description, and indicates the command option required to launch the test.

Table 4 odydiags Test and Tool Overview

Test or Tool	Description	Command Option # odydiags ***-option***
All	Runs all `odydiags` tests.	`-all`
Graphics front end (GFE)	Tests the buzz ASIC registers and the input and output formatter blocks of the GFE.	`-gfe`
SDRAM tests	Transfers host data to the SDRAMs and then back to host for comparison.	`-sdram`
DBE/PBJ tests		
DCB bus test	Tests the display control bus (DCB) that connects the buzz ASIC and the PB&J ASIC.	`-dcb`
I2C bus test	Tests the I2C data bus by sending walking 1 and 0 patterns to the phase-locked loop (PLL) register.	`-i2c`
CMAP data bus test	Checks the data bus exiting the color maps (CMAPs). *	`-cmap_data_bus`
CMAP address bus test	Checks the address bus entering the color maps. *	`-cmap_addr_bus`
CMAP pattern	Writes repeating values to CMAP memory and compares.	`-cmap_pattern`
Analog output test	Tests the three analog output lines that connect the PB&J ASIC to the monitor.	`-analog_out`
Raster-texture-shader tests	Runs all raster, texture, and shader (RTS) tests. *	`-rts`
Debug tools and utilities		
Reset graphics	Resets all the hardware on the graphics board.	`-reset`
Read a register	Enables you to read any buzz ASIC register.	`-readreg`
Write a register	Enables you to write any Odyssey register.	`-writereg`
Hardware revision information	Displays hardware revision information for the Odyssey board.	`-getinfo`
Load the CMAP	Loads various color maps.	`-load_cmap`
Read a pixel value	Enables you to read a portion of the screen.	`-readpix`
Translate (x, y) to an SDRAM chip	Enables you to enter pixel coordinates and other required information and returns a list of the SDRAMs that contain the pixel data.	`-xy_to_sdram`
SDRAM DMA write/read utility	Enables you to program a DMA operation to write to and/or read from the graphics memory (SDRAMs).	`-mem_write_read` `-dma_param_file` <*filename*>

* You must run `odydiags -reset` and `odydiags -load_cmap` before you run CMAP or RTS tests.

User Interface Description

The Odyssey diagnostics executable is `odydiags`.

The user interface uses the diagnostic command-line options that Table 2 lists. The basic command syntax is:

Example 1 Odydiags Command Line Syntax

```
# odydiags -<testname> -<various options>
```

All available tests can be run with the `-all` option:

```
# odydiags -all
```

Note: Use the `run.pl` perl script to run all of the available tests.

Tests can also be run with the `-dont_clear_flags` option. By default, the flag set register is cleared after each test. The `-dont_clear_flags` option prevents the flag set register from clearing after each test. This option is useful for debugging.

GFE Tests

The graphics front end (GFE) tests include:

- A register test
- Input/output formatter tests

You can run all the GFE tests by using the `-gfe` command line option:

Example 2 All GFE Tests

```
# odydiags -gfe
```

Using the `-gfe` option runs the register test and the input/output formatter tests. The input/output formatter tests run in the default mode without using a direct memory access (DMA) parameter file.

Refer to “Valid Values for a DMA Parameter File” on page 13 for more information about DMA parameter files.

Register Test

The register test (`-register`) writes and verifies a walking 0 and 1 pattern to various buzz ASIC registers. The command line syntax is:

Example 3 Register Test

```
# odydiags -registers
```

The following buzz ASIC registers are tested:

XT_REQ_TIMEOUT
XT_INTR_DST_ADDR_HI
XT_INTR_DST_ADDR_LO
FLCTL_CONFIG
FLAG_USER_EN_SET
FLAG_INTR_ENAB_SET
FLAG_SET_PRIV

Note: You must run the `odydiags -reset` command if this test is run after the diagnostic board manager program (`diag_brdmgr`), otherwise the system panics. This happens because the `diag_brdmgr` program clears the shadow state. The register read operation fails unless you enter the reset command first.

The `diag_brdmgr` is a program that invokes `/usr/gfx/gfxinit` and then downloads a timing table. The diagnostics require initialization by the `gfxinit` program. After initialization, a mini-board manager routine is invoked and exited in order for flow-control to be disabled. This enables the diagnostics to execute properly.

Input Formatter Test and Output Formatter Test

The input formatter test (`-input_fmt`) and the output formatter test (`-output_fmt`) share the same user interface. These tests verify the input and output formatter blocks of the GFE. Data is sent via a direct memory access (DMA) operation to the SDRAM and is formatted by the input formatter prior to being stored in the SDRAM. Data is then read out of the SDRAM and is passed through the output formatter on its way to host memory. The formatting function performed by the input and output formatters is controlled by the use of DMA parameter files (`-dma_param_file`). A sample command line for the input formatter test follows:

Example 4 Input/Output Formatter Test

```
# odydiags -input_fmt -dma_param_file ./dmafiles/if/rgba8.if
```

This command line specifies that the optional DMA parameter file to use is `./dmafiles/if/rgba8.if`. If a DMA parameter file is not specified, the test runs with the default values. The `rgba8.if` file contains the following lines:

```
transferMode    HOST_TO_SDRAM_TO_HOST
hostInFormat    B_I_INPUT_PIXEL_FORMAT_RGB
hostInData      B_I_INPUT_PIXEL_DATA_BYTE_UBYTE
gfxInFormat     ODSY_B_IO_IF_4Byte
gfxOutFormat    ODSY_B_IO_OF_RGBA8_to_RGBA8
swap            0
expansion       0
base_img_dx     4
base_img_dy     4
```

This file specifies how the DMA should be performed and the format of the pixels to be DMA'd. The transferMode of `HOST_TO_SDRAM_TO_HOST` indicates that a DMA from host memory to the SDRAM will be performed followed by a DMA from SDRAM back to host memory. The `hostInFormat` and `hostInData` specify that the pixels in host memory have a format of

RGBA-byte (4 component, 1 byte per component, for a total of 4 bytes per pixel). The `hostInFormat` and `hostInData` entries are used for host-to-SDRAM DMAs. The `gfxInFormat` and `gfxOutFormat` entries are used for SDRAM-to-host DMAs. In the example above, the `gfxInFormat` entry configures the output formatter to receive data in SDRAM stored in a 4-byte per pixel format. The `gfxOutFormat` entry configures the output formatter to perform appropriate format operations on the data that leaves SDRAM and goes to the host memory.

All input formatter DMA parameter files are located in the `./dmafiles/if/` directory. Files for the input formatter test specify a `gfxInFormat` of `ODSY_B_IO_IF_4Byte` and a `gfxOutFormat` of `ODSY_B_IO_OF_RGBA8_to_RGBA8`, which places the output formatter in pass-through mode, and output formatting does not occur. All output formatter DMA parameter files are located in the `./dmafiles/io/` directory. Files for the output formatter test specify a `hostInFormat` of `B_I_INPUT_PIXEL_FORMAT_RGB` and a `hostInData` of `B_I_INPUT_PIXEL_DATA_BYTE_UBYTE`, which places the input formatter in pass-through mode, and input formatting does not occur.

The swap and expansion lines specify whether to turn on (`1`) or off (`0`) byte swapping or expansion. The `base_img_dx` and `base_img_dy` entries specify the number of pixels to DMA in the x and y direction.

Valid Values for a DMA Parameter File

The following example lists valid entries in a DMA parameter file:

```
expansion 0=no exp or 1=exp used for L, I, LA only
swap 0=no swap or 1=swap

base_img_dx Number of pixels in the x direction

base_img_dy Number of pixels in the y direction

transferMode SDRAM_TO_HOST DMA direction
  HOST_TO_SDRAM
  HOST_TO_SDRAM_TO_HOST

[optional] do_compare 1|0, 1=compare, 0=don't compare. The default
value is 1, compare.

[optional] dma_pat 0x<pattern>, Data pattern to use for DMAs. If this
parameter is specified, this will overwrite any default pattern in the
diagnostics. For 2 byte data, the lower 16 bits are used.

[optional] origin 0x<addr>, Origin for the dma

Parameters for sdram-to-host dmas:

gfxOutFormat = how sdram read data should be formatted valid values:
  ODSY_B_IO_OF_PASS ODSY_B_IO_OF_CI8_right_to_CI8
  ODSY_B_IO_OF_I8_right_to_I8 ODSY_B_IO_OF_L8_right_to_L8
  ODSY_B_IO_OF_A8_right_to_A8 ODSY_B_IO_OF_LA8_to_A8
  ODSY_B_IO_OF_CI8_left_to_CI8 ODSY_B_IO_OF_I8_left_to_I8
  ODSY_B_IO_OF_L8_left_to_L8 ODSY_B_IO_OF_A8_left_to_A8
  ODSY_B_IO_OF_LA8_to_L8 ODSY_B_IO_OF_RGB5A1_to_RGBA8
  ODSY_B_IO_OF_I12_to_I16 ODSY_B_IO_OF_L12_to_L16
  ODSY_B_IO_OF_A12_to_A16 ODSY_B_IO_OF_RGB5A1_to_A1RGB5
  ODSY_B_IO_OF_RGB5A1_to_BGR565
  ODSY_B_IO_OF_RGB5A1_to_BGR5A1
```

```
  ODSY_B_IO_OF_RGBA4_to_ABGR4 ODSY_B_IO_OF_RGBA4_to_ARGB4
  ODSY_B_IO_OF_RGBA4_to_BGRA4
  ODSY_B_IO_OF_RGB5A1_to_A1BGR5|
  ODSY_B_IO_OF_RGB5A1_to_RGB565 ODSY_B_IO_OF_L12_to_CI12
  ODSY_B_IO_OF_L12_to_CI12 ODSY_B_IO_OF_RGB5A1_to_RGB5A1
  ODSY_B_IO_OF_RGBA4_to_RGBA4 ODSY_B_IO_OF_CI12_to_CI16
  ODSY_B_IO_OF_RGB10A2_to_RGB10A2
  ODSY_B_IO_OF_RGBA8_to_RGB8
  ODSY_B_IO_OF_RGBA8_to_ARGB8 ODSY_B_IO_OF_RGBA8_to_BGR8
  ODSY_B_IO_OF_RGBA8_to_BGRA8 ODSY_B_IO_OF_LA12_to_LA16
  ODSY_B_IO_OF_LA12_to_LA8 ODSY_B_IO_OF_YUVA8_to_YUV422_8
  ODSY_B_IO_OF_RGBA8_to_RGBA8
  ODSY_B_IO_OF_RGBA8_to_BGR565
  ODSY_B_IO_OF_RGBA8_to_RGB565
  ODSY_B_IO_OF_RGBA8_to_ABGR8
  ODSY_B_IO_OF_RGB10A2_to_A2BGR10
  ODSY_B_IO_OF_RGBA12_to_RGBA12
  ODSY_B_IO_OF_RGBA12_to_RGBA8
  ODSY_B_IO_OF_RGBA12_to_RGB8
  ODSY_B_IO_OF_RGBA12_to_A2BGR10
  ODSY_B_IO_OF_RGBA12_to_ABGR12
  ODSY_B_IO_OF_RGBA12_to_ABGR16
  ODSY_B_IO_OF_RGBA12_to_ABGR8
  ODSY_B_IO_OF_RGBA12_to_BGR8
  ODSY_B_IO_OF_RGBA12_to_RGB16
  ODSY_B_IO_OF_RGBA12_to_RGBA16
  ODSY_B_IO_OF_YUVA12_to_YUV422_16
  ODSY_B_IO_OF_RGBA16_to_RGBA16
  ODSY_B_IO_OF_RGBA16_to_RGBA8
  ODSY_B_IO_OF_RGBA16_to_A2BGR10
  ODSY_B_IO_OF_RGBA16_to_ABGR12
  ODSY_B_IO_OF_RGBA16_to_ABGR8
  ODSY_B_IO_OF_RGBA16_to_ABGR16

gfxInFormat how data is stored in sdram
  ODSY_B_IO_IF_1Byte ODSY_B_IO_IF_2Byte
  ODSY_B_IO_IF_4Byte ODSY_B_IO_IF_6Byte
  ODSY_B_IO_IF_8Byte ODSY_B_IO_IF_16Byte
  ODSY_B_IO_IF_32Byte

Parameters for host-to-sdram dmas:

hostInData
  B_I_INPUT_PIXEL_DATA_DEFAULT
  B_I_INPUT_PIXEL_DATA_BITMAP
  B_I_INPUT_PIXEL_DATA_BYTE_UBYTE
  B_I_INPUT_PIXEL_DATA_SHORT_USHORT
  B_I_INPUT_PIXEL_DATA_INT_UINT
  B_I_INPUT_PIXEL_DATA_UBYTE_3_3_2
  B_I_INPUT_PIXEL_DATA_USHORT_4_4_4_4
  B_I_INPUT_PIXEL_DATA_USHORT_5_5_5_1
  B_I_INPUT_PIXEL_DATA_UINT_10_10_10_2
  B_I_INPUT_PIXEL_DATA_U6BYTE_12_12_12_12
  B_I_INPUT_PIXEL_DATA_USHORT_5_6_5
  B_I_INPUT_PIXEL_DATA_BYTE_TO_RGBA12
  B_I_INPUT_PIXEL_DATA_USHORT_5_6_5_TO_RGBA8
```

```
hostInFormat
  B_I_INPUT_PIXEL_FORMAT_DEFAULT
  B_I_INPUT_PIXEL_FORMAT_COLOR_INDEX
  B_I_INPUT_PIXEL_FORMAT_LUMINANCE
  B_I_INPUT_PIXEL_FORMAT_INTENSITY
  B_I_INPUT_PIXEL_FORMAT_RED
  B_I_INPUT_PIXEL_FORMAT_GREEN
  B_I_INPUT_PIXEL_FORMAT_BLUE
  B_I_INPUT_PIXEL_FORMAT_ALPHA
  B_I_INPUT_PIXEL_FORMAT_LUMINANCE_ALPHA
  B_I_INPUT_PIXEL_FORMAT_RGB
  B_I_INPUT_PIXEL_FORMAT_YUV422
  B_I_INPUT_PIXEL_FORMAT_RGBA
  B_I_INPUT_PIXEL_FORMAT_ABGR
  B_I_INPUT_PIXEL_FORMAT_BGR
  B_I_INPUT_PIXEL_FORMAT_YUV422_ZEROFILL
  B_I_INPUT_PIXEL_FORMAT_COLOR_INDEX2LUMINANCE
  B_I_INPUT_PIXEL_FORMAT_ARGB
  B_I_INPUT_PIXEL_FORMAT_BGRA
```

Note: Looping on these tests (c-shell while loops) with the back-end setup causes diagnostic DMA timeouts and kernel panics after some time. Better results can be obtained if the back end is not set up. Run the `odydiags -reset` command prior to looping on these tests.

SDRAM Tests

The SDRAM is tested by using DMA operations to access data from host memory and transfer it to the SDRAM and then, using DMA operations, transfer the data from SDRAM back to host memory and compare the returned data to an expected value. All tests are performed using 2-byte per pixel data.

Because SDRAM can be quite large (32, 64, or 128 Mbytes), the SDRAM tests DMA data in chunks rather than trying to access the entire SDRAM at once (for example, a chunk might be 8 Mbytes). If a chunk is 8 Mbytes, then a test of a 32-Mbyte system would require four DMA write-read-pair operations to verify all of SDRAM.

The standard SDRAM graphics memory tests can be run by using the following command. The `-sdram` command-line argument invokes the `-cfifomem`, `-data_bus`, `-addr_bus`, and `-patterns` tests.

Example 5 Standard SDRAM Test

```
# odydiags -sdram
```

All of the SDRAM tests also can be run with the following three optional arguments:

Table 5 SDRAM Test Arguments

`-write_only`	Force the test to do only DMA writes. DMA reads are skipped.
`-read_only`	Force the test to do only DMA reads. DMA writes are skipped.
`-asyncdma`	Run the test using asynchronous DMAs. The default setting uses synchronous DMAs.

Example 6 Optional SDRAM Test Arguments

```
# odydiags -sdram -write_only
# odydiags -sdram -read_only
# odydiags -sdram -asyncdma
```

CFIFO Memory Test

Tests the area of SDRAM where the command first-in-first-out (CFIFO) data normally resides. An asynchronous DMA operation is used to test this area of SDRAM because asynchronous DMAs do not require a CFIFO.

Example 7 Cfifomem Test

```
# odydiags -cfifomem
```

Data Bus Test

Tests the integrity of data lines by using DMAs to walk 1 and 0 data patterns to the same memory address. The data bus test can also be run individually.

Example 8 Data Bus Test

```
# odydiags -data_bus
```

Address Bus Test

Tests the integrity of the address lines by using DMAs to walk 1 and 0 data patterns on the address bus. The data stored at each address is the inverse of the address. The address bus test can also be run individually:

Example 9 Address Bus Test

```
# odydiags -addr_bus
```

Note: The following tests are detailed SDRAM memory tests that are useful for debugging. Most of these tests take a long time to run.

Address Unique Test

Uses DMA operations to write the address of each memory location to that memory location. This takes a long time to run because addressing is performed on a per-tile basis. A 32-Mbyte system requires more than 65,000 DMA operations (which correspond to the 65,000 tiles).

Example 10 Address Unique Test

```
# odydiags -addr_uniq
```

Walking One Test

Fill the entire memory with data that has only 1 bit turned on. Walk the bit along the data bus.

Example 11 Walking One Test

```
# odydiags -walking_1
```

Walking Zero Test

Fill the entire memory with data that has only 1 bit turned off. Walk the 0 bit along the data bus.

Example 12 Walking Zero Test

```
# odydiags -walking_0
```

Walking Address Unique Test

The first pass is the same as the address unique test. Subsequent passes through memory apply a left-shift operation to the data until the maximum bit value (32) is reached.

Example 13 Walking Unique Test

```
# odydiags -walking_u
```

Pattern Test

Tests the entire memory using a mixed matrix pattern of data.

Example 14 Pattern Test

```
# odydiags -patterns
```

Use p1=0x5a5a, p2=0x3c3c, p3=0xf0f0, p4=0xa5a5, p5=0xc3c3, p6= 0x0f0f as data patterns. A total of 6 passes through memory occurs.

Table 6 Pattern Test Data Patterns

addr	pass 1	pass 2	pass 3	pass 4	pass 5	pass 6
a(0)	p1	p2	p3	p4	p5	p6
a(1)	p2	p3	p4	p5	p6	p1
a(2)	p3	p4	p5	p6	p1	p2
a(3)	p4	p5	p6	p1	p2	p3
a(4)	p5	p6	p1	p2	p3	p4
a(5)	p6	p1	p2	p3	p4	p5
a(6)	p1	p2	p3	p4	p5	p6

Table 6 (continued) Pattern Test Data Patterns

addr	pass 1	pass 2	pass 3	pass 4	pass 5	pass 6
a(7)	p2	p3	p4	p5	p6	p1
(pattern continues)						

DBE/PBJ Tests

DCB Bus Test

Tests the display control bus (DCB) that connects the buzz ASIC and the PB&J ASIC. A walking 1 and a walking 0 pattern are sent across the 8-bit DCB by writing to a PB&J register.

Example 15 DCB Bus Test

```
# odydiags -dcb
```

I2C Bus Test

Tests the I2C data bus by sending walking 1 and walking 0 patterns to the phase-locked loop (PLL) register.

Example 16 I2C Bus Test

```
# odydiags -i2c
```

CMAP Data Bus Test

Note: You must run `odydiags -reset` before you run any CMAP tests.

The data bus test checks the data bus exiting the CMAPs (the CMAPs can also be referred to as the CLUTs).

The XMAP pixel override mechanism is used to set all red, green, and blue pixel data entering the CMAP to 0x0. Then walking 1 and 0 patterns are written to array location 0 of the CMAP memory (cmap_clut[0]). Array cmap_clut[0] is then read for each pattern, and a CMAP CRC value is captured. The CRC value is compared to a known value, and the results of the comparison are displayed.

Example 17 CMAP Data Bus Test

```
# odydiags -cmap_data_bus
```

CMAP Address Bus Test

Note: You must run `odydiags -reset` before you run the CMAP tests.

The address bus test checks the address bus entering the CMAPs. The XMAP pixel override mechanism is used to set all red, green, and blue pixel data entering the CMAP to walking 1 and walking 0 patterns. The corresponding CMAP entries are then read, and a CMAP CRC value is captured. The CRC value is compared to a known value, and the results of the comparison are displayed.

Example 18 CMAP Address Bus Test

```
# odydiags -cmap_addr_bus
```

CMAP Test -read_cmap Argument

Note: You must run `odydiags -reset` before you run any CMAP tests.

You can use the `-read_cmap` optional argument for any CMAP test.

```
# odydiags -cmap_addr_bus -read_cmap
```

The `-read_cmap` option reads the contents of the CMAP memory that is being tested. The default setting is to rely solely on the CMAP CRC values to determine whether the tests have passed. The contents of the CMAP memory are never read. The `-read_cmap` option is useful for debugging a faulty graphics board.

CMAP Patterns Test

Note: You must run `odydiags -reset` before you run any CMAP tests.

The `-cmap_pattern` option tests the CMAPs by writing and repeating a 6-pattern set of values to the CMAP memory. The XMAP pixel override mechanism is used to increment the red, green, and blue pixel data entering the CMAP from 0 to 8191 (maximum size of the CMAPs). Each CMAP entry is then accessed and a CMAP CRC value is captured. The CRC value is compared to a known value, and the results of the comparison are displayed.

Example 19 CMAP Patterns Test

```
# odydiags -cmap_pattern
```

Analog Output Test

Tests the three analog output lines (PBJ_ANAL_RED_H, PBJ_ANAL_GRN_H, and PBJ_ANAL_BLU_H) that connect the PB and J ASIC to the monitor. This test is used only in the repair area because a monitor must be attached. The test uses the red, green, and blue comparator output bits of the DAC control register. If a failure occurs, check the PBJ_ANAL_RED_H, PBJ_ANAL_GRN_H, and PBJ_ANAL_BLU_H signals leaving the PB and J ASIC.

Example 20 Analog Output Test

```
# odydiags -analog_out
```

Raster-Texture-Shader Tests

Note: You must run `odydiags -reset` before you run any raster-texture-shader (RTS) tests.

All RTS tests have a similar syntax:

Example 21 RTS Tests

```
# odydiags -<testname>
```

In addition, the `-noverify` option can be added to disable CRC and pixel read-back and verification.

```
# odydiags -<testname> -noverify
```

The following test groupings are available:

```
# odydiags -rts
```
Run all RTS tests

```
# odydiags -raster
```
Run only raster tests (`points`, `lines`, `aa_lines`, `aa_points`, `triangles`)

```
# odydiags -shader
```
Run only shader tests (`alpha_func`, `blend_logic`)

```
# odydiags -texture
```
Run only texture tests (`checker_3d`)

Points Test

This test draws several simple points. The command syntax is:

Example 22 Points Test

```
# odydiags -points
```

Lines Test

This test draws several simple lines. The command syntax is:

Example 23 Lines Test

```
# odydiags -lines
```

Anti-aliased Points Test

This test draws anti-aliased/wide points. The command syntax is:

Example 24 Anti-aliased Points Test

```
# odydiags -aa_points
```

Anti-aliased Lines Test

This test draws anti-aliased/wide lines. The command syntax is:

Example 25 Anti-aliased Lines Test

```
# odydiags -aa_lines
```

Triangles Test

This test draws several simple triangles. The command syntax is:

Example 26 Triangles Test

```
# odydiags -triangles
```

Alpha Function Test

This test verifies the alpha function by drawing quads of varying alpha values. The command syntax is:

Example 27 Alpha Function Test

```
# odydiags -alpha_func
```

Blending Test

This test verifies the blending logic function by blending several quads together. The command syntax is:

Example 28 Blending Test

```
# odydiags -blend_logic
```

Debug Tools and Utilities

Reset Graphics

The `-reset` option resets all the hardware on the graphics board. The command syntax is:

Example 29 Reset Graphics

```
# odydiags -reset
```

Read a Register

The `-readreg` command line option enables you to read any buzz ASIC register.

Example 30 Read a Register

```
# odydiags -readreg
REV                 Odyssey Diags version 2.0 built on Jul 13 2001 at 13:09:50
INFO                Start time Thu Oct 11 13:27:51 2001
INFO                Running on IRIX64 6.5.10m 07171440 IP35 (shad)
INFO                Odydiags found and attached 1 Odyssey board(s)
BRD#                Testing Odyssey board # 0 ...
CMDL                ./odydiags -readreg
TEST readreg        Read a register                       Test(1/1), Loop(1/1)
Read a register
Note: DBE/PBJ read register range is 0x220000 - 0x23ffff
e.g. Enter 0x222580 to read dbe register 0x2580
Enter the register address to read in hex (e.g. 0x1000): 0x30
INFO                base= 0x4000, addr = 0x30, data read = 0x0
RSLT readreg        PASS         Board#0: readreg completed
#
```

Write a Register

The `-writereg` command line option enables you to write any Odyssey register.

Example 31 Write a Register

```
# odydiags -writereg
REV                 Odyssey Diags version 2.0 built on Jul 13 2001 at 13:09:50
INFO                Start time Thu Oct 11 13:29:10 2001
INFO                Running on IRIX64 6.5.10m 07171440 IP35 (shad)
INFO                Odydiags found and attached 1 Odyssey board(s)
BRD#                Testing Odyssey board # 0 ...
CMDL                ./odydiags -writereg
TEST writereg       Write a register                      Test(1/1), Loop(1/1)
Write a register
Note: DBE/PBJ write register range starts at 0x400000
e.g. Enter 0x402580 to write dbe register 0x2580
Enter the register address to write in hex (e.g. 0x1088): 0x400000
Enter the value to write in hex (e.g. 0x20): 0
INFO                base = 0x4000, addr = 0x400000,
INFO                data written = 0x3000000100000000
RSLT writereg       PASS         Board#0: writereg completed
#
```

Hardware Revision Information

The `-getinfo` command-line option displays hardware revision information for the Odyssey board:

Example 32 Hardware Revision Information

```
# odydiags -getinfo
REV                Odyssey Diags version 2.0 built on Jul 13 2001 at 13:09:50
INFO               Start time Thu Oct 11 13:30:05 2001
INFO               Running on IRIX64 6.5.10m 07171440 IP35 (shad)
INFO               Odydiags found and attached 1 Odyssey board(s)
BRD#               Testing Odyssey board # 0 ...
CMDL               ./odydiags -getinfo
TEST getinfo       Get system information                Test(1/1), Loop(1/1)
INFO               buzz revision is 10
INFO               sdram size is 128MB
INFO               sdram banks are 4
INFO               sdram casl is 3
INFO               board version is 1
INFO               pbj version is 1
RSLT getinfo       PASS         Board#0: getinfo completed
#
```

Load the CMAP

This command loads various color maps (CMAPs). The various CMAP patterns are specified by the `-pat` option. If a pattern is not specified, the command loads the standard CMAP by default. The supported CMAP patterns are:

Example 33 Loading the Color Map

odydiags -load_cmap
`-pat` not used, loads standard pattern.

odydiags -load_cmap -pat 1
Loads standard pattern.

odydiags -load_cmap -pat 2
Loads repeating ramp.

odydiags -load_cmap -pat 3
Loads 6-pattern data.

odydiags -load_cmap -pat 4
Loads address unique pattern.

odydiags -load_cmap -pat 5
Loads address bus pattern.

odydiags -load_cmap -pat <*specify pattern*>
Repeats the specified pattern.

Read a Pixel Value

The `-readpix` option enables you to read a portion of the screen. You are prompted to enter the coordinates of a rectangular screen region; then the command performs a DMA read operation on the graphics memory that corresponds to the specified region. Data is read in RGBA-byte format. A sample session follows:

```
   # odydiags -readpix
REV                Odyssey Diags version 2.0 built on Jul 13 2001 at 13:09:50
INFO               Start time Thu Oct 11 13:31:49 2001
INFO               Running on IRIX64 6.5.10m 07171440 IP35 (shad)
INFO               Odydiags found and attached 1 Odyssey board(s)
BRD#               Testing Odyssey board # 0 ...
CMDL               ./odydiags -readpix
TEST readpix       Read Pixels command                    Test(1/1), Loop(1/1)
INFO
NOTE: The upper left corner is (0,0)

This routine will allow you to read a rectangular region of pixels
Enter the x-coordinate for the lower left corner (0-1279): 0
Enter a the y-coordinate for the lower left corner (0-1023): 0
Enter the x-coordinate for the upper right corner (0-1279): 2
Enter the y-coordinate for the upper right corner (0-1023): 2
Pixel at (0, 0) red = 0x0
Pixel at (0, 0) green = 0x0
Pixel at (0, 0) blue = 0x0
Pixel at (0, 0) alpha = 0x0
Pixel at (0, 0) value = 0x0
Pixel at (1, 0) red = 0x0
Pixel at (1, 0) green = 0x0
Pixel at (1, 0) blue = 0x0
Pixel at (1, 0) alpha = 0x0
Pixel at (1, 0) value = 0x0

Pixel at (0, 1) red = 0x0
Pixel at (0, 1) green = 0x0
Pixel at (0, 1) blue = 0x0
Pixel at (0, 1) alpha = 0x0
Pixel at (0, 1) value = 0x0

Pixel at (1, 1) red = 0x0
Pixel at (1, 1) green = 0x0
Pixel at (1, 1) blue = 0x0
Pixel at (1, 1) alpha = 0x0
Pixel at (1, 1) value = 0x0
RSLT readpix       PASS                    readpix completed
LOOP               Completed Loop 1 of 1, duration: 13.463 sec PASS
```

Translate (x, y) to an SDRAM Chip

This routine enables you to enter pixel coordinates and other required information and returns a list of the SDRAM chips that contain the pixel data. The required information is:

- x coordinate and y coordinate
- pixel origin, stride and size
- field size and offset
- data
- number of memory banks and memory size

Example 34 Translate x,y to SDRAM Chip

```
    # ./odydiags -xy_to_sdram
REV                 Odyssey Diags version 2.0 built on Jul 13 2001 at 13:09:50
INFO                Start time Thu Oct 11 13:33:44 2001
INFO                Running on IRIX64 6.5.10m 07171440 IP35 (shad)
INFO                Odydiags found and attached 1 Odyssey board(s)
BRD#                Testing Odyssey board # 0 ...
CMDL                ./odydiags -xy_to_sdram
TEST xy_to_sdram    Translate (x,y) to a particular sdram  Test(1/1), Loop(1/1)
Enter the x-coordinate: 10
Enter the y-coordinate: 25
Enter the pixel origin: 0x2000
Enter the pixel stride: 0
Enter the pixel size in bytes (16bytes not supported):2
Enter the field size in bytes (16bytes not supported):2
Enter the data (must be non-zero): 0xbeef
Enter the field offset: 0
Enter the number of banks: 1
Enter the sdram size (32, 64 or 128): 32


X        Y        Origin   Stride  fbdp lin_adr  addr     sdrm_idx chip  data
0000000a 00000019 00002000 00000000 2   00000100 00000000 00000000  d    beef

                  SDRAM chip # 13 is SDRAM Bank 0_A
                  SDRAM chip # 13 is SDRAM Bank 0_A
                  More pixels to check? (yes=1, no=0): 0
```

SDRAM DMA Write/Read Utility

The `-mem_write_read` utility enables a user to program a DMA operation to write to and/or read from the graphics memory (SDRAMs). The command line syntax and sample output follow:

Example 35 SDRAM DMA Write/read Utility

```
# odydiags -mem_write_read -dma_param_file <filename>
REV                 Odyssey Diags version 2.0 built on Jul 13 2001 at 13:09:50
INFO                Start time Thu Oct 11 13:37:00 2001
INFO                Running on IRIX64 6.5.10m 07171440 IP35 (shad)
INFO                Odydiags found and attached 1 Odyssey board(s)
BRD#                Testing Odyssey board # 0 ...
CMDL                ./odydiags -mem_write_read -dma_param_file
CMDL                ./dmafiles/if/rgb8.if
INFO                dma_param_file is ./dmafiles/if/rgb8.if

TEST mem_write_read Write / Read data to graphics memory Test(1/1), Loop(1/1)
INFO transferMode= HOST_TO_SDRAM_TO_HOST, 3
INFO hostInFormat= B_I_INPUT_PIXEL_FORMAT_RGB, 11
INFO hostInData= B_I_INPUT_PIXEL_DATA_BYTE_UBYTE, 2
INFO gfxInFormat= ODSY_B_IO_IF_4Byte, 2
INFO gfxOutFormat= ODSY_B_IO_OF_RGBA8_to_RGBA8, 8
INFO swap= 0
INFO expansion= 0
INFO base_img_dx= 4
INFO base_img_dy= 4
INFO do_compare= 0
INFO dma_pat= 0xbaadcafe
INFO origin= 0x2040
INFO

RSLT mem_write_read PASS mem_write_read test PASSED file: ./dmafiles/if/rgb8.if
LOOP Completed Loop 1 of 1, duration: 5.450 sec PASS
```

Refer to “Valid Values for a DMA Parameter File” on page 13 for a detailed explanation of each option shown below. A sample DMA parameter file follows:

```
transferMode HOST_TO_SDRAM_TO_HOST
hostInFormat B_I_INPUT_PIXEL_FORMAT_RGB
hostInData B_I_INPUT_PIXEL_DATA_BYTE_UBYTE
gfxInFormat ODSY_B_IO_IF_4Byte
gfxOutFormat ODSY_B_IO_OF_RGBA8_to_RGBA8
swap 0
expansion 0
base_img_dx 4
base_img_dy 4
do_compare 0
dma_pat 0xbaadcafe
origin 0x2040
```

Error Messages

The following error messages are listed in numerical order:

`WARNING: A correctable ECC error was detected`

This is just a warning message. No action is necessary. If this message appears frequently, run the SDRAM tests with the `-asyncdma` command line option to identify a potentially bad SDRAM chip.

`WARNING: Multiple correctable ECC errors detected`

This is just a warning message. No action is necessary. If this message appears frequently, run the SDRAM tests with the `-asyncdma` switch to identify a potentially bad SDRAM chip.

`**** ERROR 000000 Can't allocate memory for base_image/base_image_rcv`

The diagnostic cannot allocate memory for DMA data.

`**** ERROR 000001 Can't open <`*name*`> file!`

The system cannot open the specified DMA parameter file, <*name*>.

`**** ERROR 000002 Unknown dma parameter: <`*name*`>`

Ensure that the DMA parameter file is correct.

`**** ERROR 000003 Can't find the error code for <`*name*`>`

The error code for <*name*> cannot be found in the error code file. Check the contents of the error code file, ERRCODES.

`**** ERROR 000004 Cannot find the correct host pixel size`

Ensure that the DMA parameter file is correct.

`**** ERROR 000006 <`*num*`>bpp dma from host to sdram failed`

The host-to-SDRAM DMA failed.

`**** ERROR 000007 <`*num*`>bpp dma from sdram to host failed`

The SDRAM-to-host DMA failed.

`**** ERROR 000008 <`*num*`>bpp dma comparison failure`

The DMA completed but the received data did not match the expected data.

`**** ERROR 000009 dma common doesn't do async dmas yet`

Asynchronous DMAs are not supported in this test.

`**** ERROR 000010 invalid bpp in odsy_tile_shift_bits: <`*num*`>`

`**** ERROR 000011 invalid bpp in odsy_sdram_addr_from_tilenr: <`*num*`>`

`**** ERROR 000012 invalid bpp in odsy_tilenr_from_sdram_addr: <`*num*`>`

The number of bytes per pixel is invalid. Ensure that the parameter file is correct.

**** ERROR 000013 getOdsyInfo: invalid sdram size: <*num*>

The SDRAM size is invalid <*num*>Mbytes. Valid sizes are 32, 64, and 128 Mbytes.

**** ERROR 000014 getOdsyInfo: invalid sdram banks: <*num*>

The specified number of SDRAM banks is invalid <*num*>.Valid sizes are 2 and 4.

**** ERROR 000015 getOdsyInfo: invalid sdram casl: <*num*>

The specified CAS latency is invalid <*num*>. Only 2 and 3 are valid values.

**** ERROR 000016 getOdsyInfo: invalid board version: <*num*>

The specified board version is invalid.

**** ERROR 000017 Error: Unable to allocate ucode memory buffer

The system cannot allocate memory for downloading the microcode file.

**** ERROR 000018 FAIL

The diagnostic cannot download the microcode.

**** ERROR 000019 Error: Can't open file <*name*>

The diagnostic cannot open the specified microcode file, <*name*>.

**** ERROR 000020 odsyDownload: invalid ucode length = <*num*>

The microcode file has an invalid length. Check the size of the microcode file.

**** ERROR 000021 odsyDownload: invalid ucode alignment, len = <*num*> is not in complete 12-byte lines

The microcode file must consist of complete 12-byte (96-bit) lines of data. Ensure that the size of the file is correct.

**** ERROR 000022 libdiag: cannot set to odsy discovery board number <*num*>

The graphics board has an invalid number, <*num*>.

**** ERROR 000023 libdiag diag_config_reset: invalid sdram size <*num*>MB. (32,64,128) are valid.

The SDRAM size is invalid <*num*>Mbytes. Valid sizes are 32, 64, and 128.

**** ERROR 000024 libdiag diag_config_reset: invalid sdram banks <*num*>. (2,4) are valid.

The specified number of SDRAM banks is invalid <*num*>. Valid numbers are 2 and 4.

**** ERROR 000025 libdiag diag_config_reset: invalid sdram cas latency <num>. (3,2) are valid.

The specified CAS latency is invalid <*num*>. Only 2 and 3 are valid values.

**** ERROR 000026 odsy diag dma failed with rc:<*num*>

The DMA failed.

**** ERROR 000027 odsy diag get shadow pointers failed with rc:<*num*>

**** ERROR 000028 odsy diag release shadow pointers failed with rc:<*num*>

**** ERROR 000029 timed out waiting for ctxsw

**** ERROR 000032 open of <*name*> failed

The diagnostic could not open the graphics device.

**** ERROR 000033 fcntl of <*name*> failed

The diagnostic could not open the graphics device.

**** ERROR 000034 can't open graphics device driver

The diagnostic could not open the graphics device.

**** ERROR 000035 can't get number of graphics boards (no gfx driver?)

The diagnostic cannot determine how many graphics boards are in the system.

**** ERROR 000036 getboard info failed.

The diagnostic cannot determine how many graphics boards are in the system.

**** ERROR 000037 getboard info (2) failed.

The diagnostic cannot determine how many graphics boards are in the system.

**** ERROR 000038 can't open the graphics device driver

The diagnostic could not open the graphics device.

**** ERROR 000039 discover boards didn't find an odyssey boards

No Odyssey graphics boards were found in the system.

**** ERROR 000040 you don't have a gfx board number to go with odsy board number <*num*>

The specified board number, <*num*>, is invalid.

**** ERROR 000041 IS_MANAGED query ioctl failed

The graphics board has already been initialized by another program.

**** ERROR 000042 board <*num*> is managed

The graphics board has already been initialized by another program.

**** ERROR 000043 couldn't set diag flag for board <*num*>

Ensure that you have superuser privileges when you run this test.

**** ERROR 000044 gfx attach board failed

Software could not attach the board to the system.

**** ERROR 000045 gfx map all failed

The diagnostic could not map the graphics board to the system address space.

**** ERROR 000046 gfx detach board failed

Software could not detach the board from the system.

**** ERROR 000047 <*num*> bpp not yet supported

Formatting with <*num*> bytes per pixel is not supported in this test. Ensure that the parameter file is correct.

**** ERROR 000048 Invalid pixel format

Ensure that the parameter file is correct.

**** ERROR 000049 Can't allocate memory for if_postFmtBfr->valid

The system cannot allocate memory for the DMA.

**** ERROR 000050 <*val*> is not a valid value for the parameter: <*name*>",

Ensure that the DMA parameter file is correct.

**** ERROR 000051 WIDGET_REQ_TIMEOUT_REG Addr 0x<*value*> exp 0x<*value*> recv 0x<*value*>

The expected data (exp) did not match the actual data (recv) when this register was read using the -registers command line option.

**** ERROR 000051 WDT_INTR_DST_HI_REG Addr 0x<value> exp 0x<value> | recv 0x<value>

The expected data (exp) did not match the actual data (recv) when this register was read using the -registers command line option.

**** ERROR 000051 WDT_INTR_DST_LO_REG Addr 0x<*value*> exp 0x<*value*> recv 0x<*value*>

The expected data (exp) did not match the actual data (recv) when this register was read using the -registers command line option.

**** ERROR 000051 GFX_FLOW_CNTL_TIMEOUT Addr 0x<*value*> exp 0x<*value*> recv 0x<*value*>

The expected data (exp) did not match the actual data (recv) when this register was read using the -registers command line option.

**** ERROR 000051 FLAG_ENAB_SET Addr 0x<*value*> exp 0x<*value*> recv 0x<*value*>

The expected data (exp) did not match the actual data (recv) when this register was read using the -registers command line option.

**** ERROR 000051 FLAG_INTR_ENAB_SET Addr 0x<*value*> exp 0x<*value*> recv 0x<*value*>

The expected data (exp) did not match the actual data (recv) when this register was read using the -registers command line option.

**** ERROR 000051 FLAG_SET_PRIV Addr 0x<*value*> exp 0x<*value*> recv 0x<*value*>

The expected data (exp) did not match the actual data (recv) when this register was read using the -registers command line option.

```
**** ERROR 000052 Exp=0x<value> vs Rcv=0x<value> @ (x, y)

INFO
Pixel location information:
X        Y        Origin   Stride   fbdp  lin_adr  addr     sdrm_idx  chip  data
00000000 00000003 00002000 00000001  4    00000080 00000000 00000000   14   0003
00000000 00000003 00002000 00000001  4    00000080 00000000 00000000   12   54ab
SDRAM chip # 12 is SDRAM Bank 2_A
SDRAM chip # 14 is SDRAM Bank 2_B
```

The expected data (`Exp`) did not match the actual data (`Rcv`) during the readback of data in SDRAM. Table 7 lists the fields in the pixel location information:

Table 7 Pixel Location Information

Field	Description
`X`	X coordinate in hexadecimal. (x = 0, y = 0) is the upper-left corner of the DMA region.
`Y`	Y coordinate in hexadecimal
`Origin`	DMA beginning address in hex. The default beginning address is 0x2000.
`Stride`	Stride in tiles. 1 tile is 16 pixels wide.
`fbdp`	Pixel size in bytes.
`lin_adr`	This field is listed for software debugging.
`addr`	This field is listed for software debugging.
`sdrm_idx`	This field is listed for software debugging.
`chip`	The chip number 0-15. This designation is for software use only.
`data`	The example shows a 32-bit dataword, the upper 16 bits contain the data 0003 and are found in chip 12; the lower 16 bits contain the data 54ab and are found in chip 14.

The chip number is translated to a specific SDRAM chip on the graphics board. In the example above, the following lines indicate which SDRAM chips are failing:

```
SDRAM chip # 12 is SDRAM Bank 2_A
SDRAM chip # 14 is SDRAM Bank 2_B
```

**** ERROR 001000 Expected data = 0x<*value*>, Received data = 0x<*value*>, diff = 0x<*value*>

Refer to the description for error number 000052.

**** ERROR 001001 DMA operation failed

The DMA failed during the SDRAM tests.

**** ERROR 002001 Microcode Memory word <*num*>, exp=<*exp*> rcv=<*rcv*> diff=<*diff*>

The data that was read back, <*rcv*>, from the microcode RAM at line <*num*> did not match what was written, <*exp*>. Run the test using a different data pattern to determine whether the data pattern caused the failure.

**** ERROR 002002 Unknown pattern, <*num*>. Valid values for -pat are 1 and 2

The value specified with the -pat option is not valid. Choose another value.

**** ERROR 003000 Could not find the key, <*val*> in <*filename*>

The CMAP diagnostics could not locate the string <*val*> in the file, <*filename*>. Open <*filename*> to ensure that the string, <*val*>, is found in the file.

**** ERROR 003001 Red channel cmap crc miscompare: exp=<*val*>, rcv=<*val*>

The CMAP diagnostics discovered a CRC miscompare. Run the test again with the -use_pixel_input switch. If the test fails again, then the PB&J ASIC is defective. If the test passes when you use the -use_pixel_input option, then the PB&J ASIC is working, but the diagnostic path is defective.

**** ERROR 003002 Green channel cmap crc miscompare: exp=<*val*>, rcv=<*val*>

The CMAP diagnostics discovered a CRC miscompare. Run the test again with the -use_pixel_input switch. If the test fails again, then the PB&J ASIC is defective. If the test passes when you use the -use_pixel_input option, then the PB&J ASIC is working, but the diagnostic path is defective.

**** ERROR 003003 Blue channel cmap crc miscompare: exp=<val>, rcv=<val>

The CMAP diagnostics discovered a CRC miscompare. Run the test again with the -use_pixel_input option. If the test fails again, then the PB&J ASIC is defective. If the test passes when you use the -use_pixel_input option, then the PB&J ASIC is working, but the diagnostic path is defective.

**** ERROR 003004 Cmap entry <addr> exp=<val> rcv=<val>

The expected value of the CMAP at location <*addr*> was not what was expected. The PB&J ASIC is defective.

**** ERROR 003005 odsy: hw write error during timing table download

A hardware write error occurred while downloading the timing table.

**** ERROR 003006 <*register name*> exp <*val*> recv <*val*>

The read value did not match the expected value when this register was read.

**** ERROR 003007 Timed out waiting for i2c idle or wait, status = <*status*>

The I2C test timed out waiting for the PLL to go idle. Check the PLL clock.

**** ERROR 003008 Timed out waiting for dbe fifo

The DBE FIFO did not empty.

**** ERROR 003009 CMAP load error

The CMAP could not be loaded with data.

**** ERROR 003010 Red comparator: exp <*val*> but rcv <*val*>

The analog output test indicates an error on the PBJ_ANAL_RED_H signal.

**** ERROR 003011 Green comparator: exp <*val*> but rcv <*val*>

The analog output test indicates an error on the PBJ_ANAL_GRN_H signal.

**** ERROR 003012 Blue comparator: exp <*val*> but rcv <*val*>

The analog output test indicates an error on the PBJ_ANAL_BLU_H signal.

**** ERROR 007000 Cannot open file <*name*>

The diagnostic cannot open the file. Ensure that the file exists.

**** ERROR 007001 RSLT errfile FAIL errorcode==ERRFILE

The diagnostic cannot read the error code file.

**** ERROR 007002 RSLT detach FAIL errorcode==DETACH

The diagnostic cannot detach the graphics board from the system.

**** ERROR 007003 Can't find the error code for <*name*>

The diagnostic cannot find the error code for <*name*> in the error code file. Check the error code file.

**** ERROR 007004 RSLT attach FAIL errorcode==ATTACH

The diagnostic cannot attach the graphics board to the system.

**** ERROR 007005 Can't fstat file <*name*>

The diagnostic cannot get the file status. Ensure that the file is correct.

**** ERROR 007006 Can't mmap file <*name*>

The diagnostic cannot memory map the file. This is a system memory problem.

**** ERROR 007007 GFX_DOWNLOAD failed

The GFX_DOWNLOAD IOCTL call (I/O control software routine) failed. This is a system problem.

**** ERROR 007008 Uncorrectable ECC error detected

An uncorrectable ECC error was detected in the command FIFO (CFIFO). Try running the SDRAM tests with the `-asyncdma` option to find the defective SDRAM chip.

**** ERROR 007009 Timed out waiting for CFIFO level to drop below <*num*>

The CFIFO did not drop below <*num*> words.

**** ERROR 007010 Timed out waiting for DFIFO level to drop below <*num*>

The DBE FIFO did not drop below <*num*> words.

**** ERROR 008000 Error: Unable to allocate memory buffer

The system cannot allocate memory for this command.

**** ERROR 008001 FAIL

The command has failed.

**** ERROR 008002 Cannot read the video format

An error occurred while reading the timing table file. Ensure that the timing table file exists.

**** ERROR 008003 Unable to get <*field name*> information from the timing table

The <*field name*> variable could not be found in the timing table file. Ensure that the timing table file is valid.

**** ERROR 008004 Couldn't get frametable information

The frame table field could not be found in the timing table file. Ensure that the timing table file is valid.

**** ERROR 008005 Couldn't get linetable information

The line table field could not be found in the timing table file. Ensure that the timing table file is valid.

**** ERROR 008006 Failed to reset board 0

The board could not be reset successfully.

**** ERROR 008007 odsyBlankScreen: ODSY_BRDMGR_SET_BLANKING failed

The call to the odsyBlankScreen IOCTL failed.

**** ERROR 008008 odsyLoadTimingTable: ODSY_LOAD_TIMING_TABLE Failed

The call to the odsyLoadTimingTable IOCTL failed.

**** ERROR 009001 Error: Can't open file <*name*>

The diagnostic cannot read the file, <*name*>.

**** ERROR 009002 FAIL

Reading the file failed.

**** ERROR 009003 CRC sleep timer failed

The system timer failed.

**** ERROR 009004 Error: Can't find name <*name*> in file: <*file*>

The string <*name*> cannot be found in the file <*file*>. Check <*file*>.

**** ERROR 009005 <CRCname> miscompare, exp = 0x<*exp*>, rcv = 0x<*rcv*>

The received value <*rcv*> for the CRC, <*CRCname*>, did not match the expected value, <*exp*>.

**** ERROR 009006 Checksum miscompare: Exp= 0x<*exp*>, Rcv= 0x<*rcv*>

The framebuffer checksum did not match the expected value.

```
**** ERROR 009007 Error at y: <num>, x: <num>, comp: <num>,
Exp= 0x<exp>, Rcv= 0x<rcv>
INFO comps 0-1 are in the first chip listed below,
comp 2-3 are INFO in the 2nd chip listed

X         Y         Origin   Stride  fbdp  lin_adr   addr      sdrm_idx  chip  data
00000000  00000003 00002000 00000001 4    00000080  00000000   00000000  14   0003
00000000  00000003 00002000 00000001 4    00000080  00000000   00000000  12   54ab
SDRAM chip # 12 is SDRAM Bank 2_A
SDRAM chip # 14 is SDRAM Bank 2_B
```

A pixel comparison error was found at the specified x,y coordinates. The term *comp* refers to the RGBA component, so comp = 0 is red, 1 = green, 2 = blue, and 3 = alpha. For a 4-byte pixel, the upper 16 bits are stored in one SDRAM chip and the lower 16 bits are stored in another SDRAM chip. Comps 0-1 are in the first chip listed, which, in the example above, is chip 14. Comps 2-3 are in the second chip listed, which is chip number 12. In the example above, if an error says it occurred in comp numbers 0 or 1, then you would check chip number 14. If the error was in comp numbers 2 or 3, you would check chip number 12.

Refer to the explanation for error number 000052 for detailed descriptions of the various fields listed.

**** ERROR 009008 TEST FAILED: <*num*> multi-bit errors found, 0 allowed.

**** ERROR 009009 TEST FAILED: <*num*> single-bit errors found, <*num*> allowed.

**** ERROR 009010 Error: Unable to allocate memory buffer

The system could not allocate memory for the pixel read buffer.

**** ERROR 009011 ERROR. Unknown error handle method:<*num*>

The diagnostics encountered an unexpected software error.

**** ERROR 009012 The SDRAM checksum is correct but the CRCs failed.

The pixel checksum is correct but the backend CRCs are not correct.

**** ERROR 009013 GL_INVALID_ENUM

The parameter specified is unknown. This is an unexpected software error.

**** ERROR 009014 diaglLineWidth: GL_INVALID_VALUE. Must be > zero

**** ERROR 009015 diaglPointSize: GL_INVALID_VALUE. Must be > zero

**** ERROR 009016 Unknown alpha function: <*num*>

**** ERROR 009017 diaglTexEnvi: Only GL_TEXTURE_ENV_MODE is allowed as the second argument

**** ERROR 009018 diaglTexEnvi: Only GL_TEXTURE_ENV is allowed as the first argument

`**** ERROR 009019 Unknown value <num>`

The parameter specified is unknown. This is an unexpected software error.

`**** ERROR 009020 Unknown primitive: <name>`

Ensure that the user specified data file is correct.

`**** ERROR 009021 glVertex2i must follow glColor4i`

Ensure that the user specified data file is correct.

`**** ERROR 009022 Unrecognized command: <name>`

Ensure that the user specified data file is correct.

Error Codes

The following Odyssey diagnostic error codes are listed alphabetically, followed by the full message that will appear on the screen. A brief description of the error code is included.

ANALOGOUT

`RSLT analog_out FAIL errorcode==ANALOGOUT`

The analog output test failed.

ATTACH

`**** ERROR 007004 RSLT attach FAIL errorcode==ATTACH`

The diagnostic cannot attach the graphics board to the system.

BADVAL

`RSLT badval FAIL errorcode==BADVAL`

The specified value is not valid. Run the test again with another value.

CFIFOMEM

`RSLT cfifomem FAIL errorcode==CFIFOMEM`

The CFIFO Memory test failed.

CURSORP

`RSLT cursorp FAIL errorcode==CURSORP`

The cursor position test failed.

DCBBUS

`RSLT dcbbus FAIL errorcode==DCBBUS`

The DCB test failed.

DETACH

```
**** ERROR 007002 RSLT detach FAIL errorcode==DETACH
```

The diagnostic cannot detach the graphics board from the system.

DMA_SW

RSLT <*multiple tests*> FAIL errcode==DMA_SW

This error indicates that the diagnostic software encountered an unexpected error. Contact the diagnostics team.

ERRFILE

```
**** ERROR 007001 RSLT errfile FAIL errorcode==ERRFILE
```

The diagnostic cannot read the error code file.

I2CBUS

```
RSLT i2c_test FAIL errcode==I2CBUS
```

The I2C Data Bus test failed.

IN_FMT

```
RSLT input_fmt FAIL errcode==IN_FMT
```

The input_fmt test failed.

MEMWRRD

```
RSLT mem_write_read FAIL errcode==MEMWRRD
```

The DMA transfer failed.

OUT_FMT

```
RSLT output_fmt FAIL errcode==OUT_FMT
```

The output_fmt test failed.

RASAALN

```
RSLT aa_lines FAIL errcode==RASAALN
```

The aa_lines test failed.

RASAAPT

```
RSLT aa_points FAIL errcode==RASAAPT
```

The aa_points test failed.

RASLINE

```
RSLT lines FAIL errcode==RASLINE
```

The lines test failed.

RASPTS

```
RSLT points FAIL errcode==RASPTS
```

The points test failed.

RASTRI

```
RSLT triangles FAIL errcode==RASTRI
```

The triangles test failed.

REGTEST

```
RSLT odsy_reg_test FAIL errcode==REGTEST
```

The register test failed.

SDRAMAB

```
RSLT addr_bus FAIL errorcode=SDRAMAB
```

The address bus test on the SDRAM has failed.

SDRAMAU

```
RSLT addr_uniq FAIL errorcode=SDRAMAU
```

The address uniqueness test on the SDRAM has failed.

SDRAMDB

```
RSLT data_bus FAIL errorcode=SDRAMDB
```

The data bus test on the SDRAM has failed.

SDRAMPT

```
RSLT patterns FAIL errorcode=SDRAMPT
```

The pattern test on the SDRAM has failed.

SDRAMW0

```
RSLT walking_0 FAIL errorcode=SDRAMW0
```

The walking 0 test on the SDRAM has failed.

SDRAMW1

```
RSLT walking_1 FAIL errorcode=SDRAMW1
```

The walking 1 test on the SDRAM has failed.

SDRAMWU

```
RSLT walking_uniqueness FAIL errorcode=SDRAMWU
```

The walking uniqueness test on the SDRAM has failed.

SHDALFA

```
RSLT alpha_func FAIL errcode==SHDALFA
```

The alpha_func test failed.

SHDBLEN

```
RSLT blend_logic FAIL errcode==SHDBLEN
```

The blend_logic test failed.

TEX3D

```
RSLT checker_3d FAIL errcode==TEX3D
```

The checker_3d test failed.

UCODEMEM

```
RSLT ucodemem FAIL errorcode=UCODEMEM
```

The microcode memory (RAM) test has failed.

USRDRAW

```
RSLT usrdraw FAIL errorcode==USRDRAW
```

The user draw command failed.

VTGDNLD

```
RSLT vtg_download FAIL errorcode=VTGDNLD
```

The VTG timing table download has failed.

Glossary

ASIC

Application-specific integrated circuit.

Buzz ASIC

A single-chip RealityEngine graphics subsystem. The buzz ASIC is a 0.25-micron CMOS chip that operates at 266 MHz and contains 1.5 million bits of on-chip SRAM.

CFIFO

Command first-in-first-out.

CLUT

The color look-up table; also referred to as the color map (CMAP). The CLUT maps color index (CI) pixels into red, green, blue (RGB) values and maps a gamma correction factor onto true color RGB pixels.

CMAP

Color map.

CMOS

Complementary metal oxide semiconductor.

CRC

Cyclic-redundancy checking. An error-detection scheme that uses parity bits generated by polynomial encoding of digital signals, appends those parity bits to the digital signal, and uses decoding algorithms that detect errors in the received digital signal.

DAC

Digital-to-analog converter.

DBE

Double-bit error.

DCB

Display control bus. An 8-bit wide bidirectional data bus that facilitates data transfer between the buzz ASIC and the PB&J ASIC.

DMA

Direct memory access.

ECC

Error-correction code.

FIFO

First-in-first-out.

GFE

Graphics front end.

IOCTL

I/O control software routine.

Odyssey graphics

Odyssey graphics is the VPro graphics subsystem. It consists of an SGI proprietary chip set that comprises the buzz ASIC, PB&J ASIC, associated SDRAM, and software.

PB&J ASIC

Pixel blaster and jammer ASIC. Display back-end chip for the Odyssey graphics subsystem.

PLL

Phase-locked loop. An electronic circuit that controls an oscillator so that it maintains a constant phase angle relative to a reference signal.

RTS tests

Raster-texture-shader tests.

SDRAM

Synchronous dynamic random-access memory. A form of DRAM that adds a separate clock signal to the control signals.

XMAP

The XMAP block changes pixels from the frame-buffer format into CLUT addresses (red, green, and blue components).

Reader Comment Form

Title: VPro™ Odyssey Graphics Diagnostic Reference **Number: 108-0314-003**

Your feedback on this publication will help us provide better documentation in the future. Please take a moment to answer the few questions below.

For what purpose did you primarily use this document?

_____Troubleshooting _____Tutorial or introduction
_____Reference information _____Classroom use
_____Other - please explain

_____

Using a scale from 1 (poor) to 10 (excellent), please rate this document on the following criteria and explain your ratings:

_____Accuracy _____
_____Organization _____
_____Readability _____
_____Physical qualities (binding, printing, page layout) _____
_____Amount of diagrams and photos _____
_____Quality of diagrams and photos _____

Completeness (Check one and explain your answer)

_____Too much information _____ Too little information _____Correct amount

_____
_____
_____

You may write additional comments in the space below. Mail your comments to the address below, fax them to us at +1 715 726 4353, or e-mail them to us at *spt@sgi.com.* When possible, please give specific page and paragraph references. We will respond to your comments in writing within 48 hours.

_____
_____
_____

NAME _____
JOB TITLE _____
E-MAIL ADDRESS _____
SITE/LOCATION _____
TELEPHONE _____
DATE _____

[or attach your business card]

Attn: Service Publications and Training
890 Industrial Boulevard
P.O. Box 4000
Chippewa Falls, WI 54729-0078
USA